KLARQUIST SPARKMAN, LLP
16th Floor World Trade Center, 121 S.W. Salmon Street, Portland, Oregon 97204 U.S.A.
PHONE: 503-226-7391 FAX: 503-228-9446

November 16, 2009

In re application of: Grover et al.
Application No. 10/628,054
Filed: July 25, 2003
Confirmation No. 4135
For: SOFTWARE DEVELOPMENT INFRASTRUCTURE
Examiner: Isaac Tuku Tecklu
Art Unit: 2192
Attorney Reference No. 3382-65598-01

PROPOSED CLAIM AMENDMENTS

1. Applicants present the attached proposed claim amendments for discussion and consideration by the Examiner. Applicants authorize the Examiner to enter the attached proposed claim amendments if they would place the Application in condition for allowance.

2. In the attached proposed amendments, independent claims 1, 15, 31, 34, and 43 have generally been amended based on language from claim 15 (specifically, the "creating … integrating … compiling … " language). However, Applicants note that the independent claims are each distinct and may use language not identically appearing in claim 15. In addition, the following amendments have been made: independent claim 34 incorporates language from dependent claims 35-37, dependent claims 4, 5, 28, and 32 have been amended, and claims 35-37 and 39-41 have been canceled.

3. Regarding the claim amendments, see the language of claims 1 and 15, and in addition see the Application at, for example, at page 7, line 25 to page 8, line 7 and page 37, line 8 to page 43, line 27, and Figs. 14-19.

Cory A. Jones
Registration No. 55,307

Proposed Claim Amendments

1. (**Currently Amended**) One or more computer-readable media with computer-executable instructions for implementing a software development architecture comprising:

a software development scenario-independent intermediate representation format;

one or more exception handling models operable to support a plurality of programming language specific exception handling models for a plurality of different source languages;

a type system operable to represent the type representations of the plurality of different source languages; and

a code generator operable to generate code targeted for a plurality of execution architectures;

wherein the code generator constructs one or more software development components of a plurality of different software development tools using the software development scenario-independent intermediate representation format, the one or more exception handling models operable to support the plurality of programming language specific exception handling models for the plurality of different source languages, and the type system operable to represent the plurality of different source languages; [[and]]

wherein the code generator further integrates the one or more software development components of the plurality of different software development tools into a software development scenario-independent framework; and

wherein the code generator further creates the plurality of different software development tools ~~using~~ by compiling the one or more software development components and the software development ~~architecture~~ scenario-independent framework.

2. (Previously Presented) The one or more computer-readable media of claim 1 wherein the architecture is scalable to produce target software development tools ranging from lightweight just-in-time (JIT) compilers to whole program optimizing compilers.

3. (Original) The one or more computer-readable media of claim 1 wherein the architecture can be configured to produce a target software development tool with varying ranges of memory footprint, compilation speed, and optimization.

4. (**Currently Amended**) The one or more computer-readable media of claim 1 wherein the software development architecture is operable to produce a software development tool modifiable by combining a modification component with the software development ~~architecture~~ scenario-independent framework.

5. (**Currently Amended**) The one or more computer-readable media of claim 1 wherein the software development architecture is operable to produce a software development tool by dynamically linking a binary version of the software development ~~architecture~~ scenario-independent framework to a modification component.

6. (Original) The one or more computer-readable media of claim 1 wherein the intermediate representation format is extensible at runtime of a software tool employing the intermediate representation format.

7. (Previously Presented) The one or more computer-readable media of claim 1 wherein the architecture is combinable with the one or more software development components.

8. (Original) The one or more computer-readable media of claim 7 wherein the one or more software development components comprise data describing a target software development tool.

9. (Original) The one or more computer-readable media of claim 7 wherein the one or more software development components provides target execution architecture data to the code generator.

10. (Original) The one or more computer-readable media of claim 7 wherein the one or more software development components provide one or more type-checking rules to the type system.

11. (Original) The one or more computer-readable media of claim 7 wherein one or more software development components provide a set of class extension declarations to the architecture.

12. (Canceled)

13. (Previously Presented) The one or more computer-readable media of claim 7 wherein the plurality of different software development tools comprises a native compiler.

14. (Previously Presented) The one or more computer-readable media of claim 7 wherein the plurality of different software development tools comprises a JIT compiler.

15. (**Currently Amended**) A method of creating a plurality of different target software development ~~tool~~ tools, the method comprising:

receiving at least one computer-readable specification specifying functionality specific to one or more software development scenarios, wherein the at least one computer-readable specification specifies the following software development scenario functionality of the plurality of different target software development ~~tool~~ tools:

target processor execution architecture;

type checking rule set;

managed execution environment;

input programming language or input binary format; and

compilation type;

creating at least one software development component for the plurality of different software development ~~tool~~ tools from the at least one specification;

integrating the at least one software development component for the plurality of different software development ~~tool~~ tools into a software development scenario-independent framework; and

compiling, at least in part, the at least one software development component and framework to create the plurality of different target software development ~~tool~~ tools;

wherein the computer-readable specification comprises functionality for processing an intermediate representation format capable of representing a plurality of different programming languages; and

wherein the intermediate representation format comprises one or more exception handling models capable of supporting a plurality of programming language-specific exception handling models for the plurality of different programming languages.

16. (Canceled)

17. (Original) The method of claim 15 wherein software development components created from a plurality of computer-readable specifications for a plurality of respective software development scenarios are integrated into the framework.

18-21. (Canceled)

22. (Original) The method of claim 15 wherein the computer-readable specification comprises one or more rulesets for type-checking one or more languages.

23. (Original) The method of claim 15 wherein the computer-readable specification comprises a set of class extension declarations specific to one or more of the software development scenarios.

24. (Canceled)

25. (Canceled)

26. (Previously Presented) The method of claim 15 wherein the intermediate representation comprises type representations capable of representing the type representations of the plurality of different programming languages.

27. (Original) The method of claim 15 further comprising:

integrating custom code specific to one of the software development scenarios.

28. (**Currently Amended**) The method of claim 15 wherein the plurality of different software development ~~tool~~ tools comprises [[one]] at least two of the group consisting of: a native compiler, a JIT compiler, an analysis tool, and a compiler development kit (CDK).

29. (Original) The method of claim 15 wherein the computer-readable specification specifies functionality of one of the group consisting of: a Pre-JIT compiler functionality, optimizer functionality, and defect detection tool functionality.

30. (Original) One or more computer-readable media containing one or more computer-executable instructions for performing the method of claim 15.

31. (**Currently Amended**) A method of creating a plurality of different target software development ~~tool~~ tools from a common framework, the method comprising:

configuring the common framework based on one or more characteristics of the plurality of different target software development ~~tool~~ tools;

~~integrating~~ creating software development components comprising one or more characteristics of the plurality of different target software development ~~tool~~ tools [[into]] from the common framework; [[and]]

integrating the software development components into a software development scenario-independent framework; and

creating the plurality of different target software development ~~tool~~ tools ~~from~~ by compiling the software development components and the ~~integrated common~~ software development scenario-independent framework;

wherein the one or more characteristics comprises an input language chosen from a plurality of different programming languages supported by the common framework for the plurality of different target software development ~~tool~~ tools; and

wherein the common framework comprises exception handling models capable of supporting a plurality of programming language-specific exception handling models for the plurality of different programming languages.

32. (**Currently Amended**) The method of claim 31 wherein the one or more characteristics can further comprise the amount of memory necessary for the plurality of different target software development ~~tool~~ tools to execute on a target architecture, the speed at which the plurality of different target software development ~~tool~~ tools will execute on a target architecture, a input binary format for the plurality of different target software development ~~tool~~ tools, or the target architecture for the plurality of different target software development ~~tool~~ tools ~~to execute on a target architecture~~.

33. (Canceled)

34. (**Currently Amended**) A method of producing a plurality of different inter-compatible software development tools, the method comprising:

receiving a software development architecture that is operable to support a plurality of different programming languages, wherein the software development architecture further comprises:

classes that are extensible through a set of declarations;

functionality for an intermediate representation format used by both the first and second software development tools; and

functionality for a type system used by both the first and second software development tools;

creating software development components for the plurality of different software development tools from the software development architecture;

creating a first software development tool by integrating the software development components into a ~~software development architecture that is operable to support a plurality of~~

~~different programming languages~~ software development scenario-independent framework and compiling the software development components and the software development scenario-independent framework; and

creating a second different software development tool based on the first software development tool and the software development components, wherein the second software development tool dynamically links to a binary version of the ~~software development architecture~~ software development scenario-independent framework;

wherein the software development architecture comprises functionality for exception handling models operable to support programming-language specific exception handling models for the plurality of different programming languages, and the software development architecture is used by both the first and second software development tools.

35-37. (**Canceled**)

38. (Canceled)

39-41. (**Canceled**)

42. (Canceled)

43. (**Currently Amended**) A method of creating a plurality of different software development ~~tool~~ tools, the method comprising:

receiving at least one computer-executable file comprising:

an intermediate representation capable of representing a plurality of different programming languages and computer executable images;

one or more exception handling models capable of supporting a plurality of programming language specific exception handling models for the plurality of different programming languages;

a type system capable of representing the type representations of a plurality of source languages; and

a code generator capable of generating code targeted for a plurality of execution architectures;

creating one or more software development components of a plurality of different software development tools from the at least one computer-executable file;

~~linking a~~ integrating the one or more software development ~~component~~ components of the plurality of different software development tools ~~to the at least one computer-executable file~~ into a software development scenario-independent framework using at least one class extension ~~declarations~~ declaration; and

creating the plurality of different software development ~~tool~~ tools [[via]] by compiling, at least in part, the linked one or more software development ~~component~~ components and ~~computer-executable file~~ the software development scenario-independent framework.